基于二次插值的天牛须搜索算法 *

廖列法†，欧阳宗英

(江西理工大学, 信息工程学院, 江西 赣州 341000)

摘 要：针对天牛须搜索算法在高维空间中搜索精度低和易陷入局部最优的问题进行了研究，提出一种新的天牛须优化算法-基于二次插值的天牛须搜索算法，称之为 QIBAS。算法在天牛进行移动后，将天牛当前位置左右两触须作为插值坐标点，利用二次插值生成一个新的解，再对比插值产生的解与当前最优解、全局最优解的适应度值，更新全局最优解。对多个单峰函数和多峰函数进行数值仿真测试，其维度分别取 100、500、1000、5000、10000，仿真结果表明，引入二次插值有效提升了 BAS 算法跳出局部最优的能力。QIBAS 在求解最优值时，其求解精度有极大的提升，收敛速度也有较明显提升，改进算法的有效性得以验证。

关键词：天牛须搜索算法；二次插值；高维空间；全局最优；收敛速度

中图分类号：TP18　　doi: 10.19734/j.issn.1001-3695.2020.03.0052

Beetle antennae search based on quadratic interpolation

Liao Liefa†, Ouyang Zongying

(*Jiangxi University of Science & Technology*, *School of Information Engineering*, *Ganzhou Jiangxi* 341000, *China*)

Abstract: Focus on the problems that low-precision and easily to fall into local optimum result in the high-dimensional space by the longhorn search algorithm, we proposed a new optimization algorithm of the short-lived beetles-the short-lived search algorithm based on quadratic interpolation which is called QIBAS. The algorithm takes tentacle's left and right tentacles as interpolation coordinate points after the beetle moved. And generate new solution with a second interpolation, then compare to the current optimal solution and the global optimal solution with the fitness of interpolation solution. At the same time to update the global optimal solution. Numerical simulation tests were performed on multiple unimodal and multimodal functions, and their dimensions take as 100, 500, 1000, 5000 and 10000. The simulation results show that the introduction of quadratic interpolation effectively improves the ability of the BAS algorithm to jump out of the local optimum. When QIBAS solves the optimal value, its solution accuracy is greatly improved, and the convergence speed is also significantly improved. The effectiveness of the improved algorithm is verified.

Key words: beetle antennae search; quadratic interpolation; high-dimensional space; global optimal; convergence speed

0 引言

群智能优化算法是人类对大自然的探索而形成的，对自然界各类生物行为的仿真模拟。智能算法广泛应用在图像处理、工业控制、生产调度、模式识别等诸多领域。智能算法的种类也颇多，有蚁群算法[1]、粒子群算法[2]、遗传算法[3]、人工蜂群算法[4]、鲸鱼算法[5]等等。在处理优化问题时，这些算法都有一定缺陷，如蚁群算法所需设置的参数较为复杂,且计算量大；粒子群算法在处理问题时易早熟收敛，尤其是复杂多峰问题；遗传算法需对问题进行编码和解码，其复杂度较高。国内外众多学者也针对这些问题对算法进行优化[6~10]。

天牛须搜索算法(beetle antennae search，BAS)[12]是 Jiang 和 Li 在 2017 年在提出的新型智能优化算法。该算法模拟天牛觅食行为，以此实现对复杂优化问题的寻优求解。BAS 算法广泛应用于各领域。如文献[13]将 BAS 算法应用于微电网多目标能量管理中，为微电网的能源优化管理和调度提供了新的解决方案；文献[14]用 BAS 优化神经网络权值和阈值，来预测喷射灌浆混凝土的无侧限抗压强度。但针对 BAS 算法易陷入局部最优、稳定性不足以及仅能解决单目标优化问题[15]的缺点，学者们提出了一系列的改进方法。如文献[16]受群体优化算法启发，将群体引入 BAS 算法中，提出了天牛群优化算法(beetle swarm antennae search，BSAS)，BSAS 克服了 BAS 随机种子导致的不同优化结果和优化精度低的问题。文献[17]将模拟退火的蒙特卡洛准则引入 BAS 中，加快算法跳出局部最优，并将优化后的算法应用于分布式电源选址定容；文献[18]将 BAS 与粒子群算法相结合，提升了其收敛速度和搜索精度。文献[19]结合 BAS 和梯度下降，选择梯度下降方向或天牛"左右须"最优解方向进行迭代。

上述对 BAS 算法的改进均取得一定效果，但对于 BAS 算法在高维空间中搜索精度低的情况，上述算法均未提出解决方案。经研究发现，在高维空间中甚至出现天牛个体不移动的情况，这大大降低了算法的寻优精度。而针对该问题，本文将二次插值引入到 BAS 算法中。在天牛进行一次移动后，借助天牛的左右两须，利用二次插值法产生新的解，再对比新解与当前全局最优解，以此更新全局最优解。仿真结果表明，改进的算法(QIBAS)在求解高维问题时相较于天牛须算法(BAS)、粒子群算法(PSO)和遗传算法(GA)，在求解精度上有了进一步的提升。

1 天牛须搜索算法

天牛须搜索算法区别于其他优化算法，该算法只包含单个个体，不需要清楚具体函数和梯度信息就能进行寻优，因此该算法运算量小、寻优速度快且易于实现。

1.1 基本原理

天牛须搜索算法的仿生机理是，天牛在觅食时不清楚食

收稿日期：2020-03-21；**修回日期：**2020-05-13　**基金项目：**国家自然科学基金资助项目(71761018，71462018)

作者简介：廖列法(1975-)，男(通信作者)，江西玉山人，教授，硕导，博士，主要研究方向为智能计算(liaolf@126.com)；欧阳宗英(1995-)，男，江西九江人，硕士研究生，主要研究方向为算法优化，智能计算.

物的具体位置，利用头上一对触须来感应食物气味的浓度，如果左触须感受到的气味浓度高，天牛朝左移动，反之则朝右移动，最后寻找到食物的确切位置。图 1 为天牛搜索食物的过程。

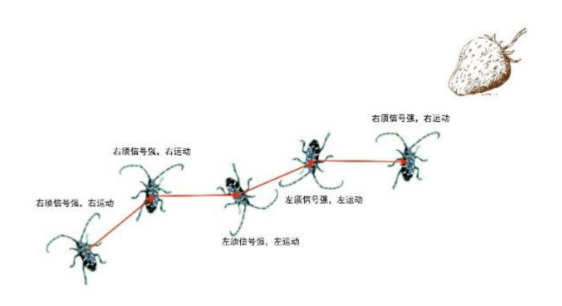


图 1 天牛搜索食物过程

Fig. 1　Beetle searching for food

天牛须搜索算法的智能机理是，单个天牛在搜索食物时，食物气味浓度即为适应度函数，这个函数在空间的不同位置有不同的值，而天牛两须分别采集自身附近两点的气味值，对比二者气味浓度，取最优值，如此迭代，知道左右两须所探测的气味浓度相差满足一定精度则停止迭代，所找到的中心点则为空间食物浓度最高点，即食物所在地。

天牛搜寻食物的过程就是天牛须算法的寻优过程。分为如下几个步骤：

a) 天牛初始朝向创建

$$\vec{d}=\frac{rands(K,1)}{\|rands(K,1)\|} \tag{1}$$

其中 $rands(\cdot)$ 为随机函数； K 为空间维度。

设置步长因子 ξ 。天牛的区域搜索能力由 ξ 决定，为了提高天牛的搜索范围，本文选取了较大初始步长，并线性方式递减步长。公式如下：

$$\xi_{t+1}=\xi_t*eta \quad (t=1,\cdots,n) \tag{2}$$

其中 eta 为递减因子，范围为 $[0,1]$ ， t 为当前迭代次数， n 为总迭代次数。

b) 建立天牛两须位置坐标

$$\begin{cases} x_l=x_t-d_0\dfrac{\vec{d}}{2} \\ x_r=x_t+d_0\dfrac{\vec{d}}{2} \end{cases} \quad （t=1,2\cdots n） \tag{3}$$

其中： x_l 表示左须的位置坐标； x_r 表示左须的位置坐标； x_t 表示迭代次数为 t 时天牛的质心坐标； d_0 为两触须之间长度，其值应足够大以覆盖适当的搜索区域，以便在开始时跳出局部最小点。

计算适应度函数 $f(\cdot)$ ， $f(\cdot)$ 指的是左右须气味浓度，即 $f(x_l)$ 和 $f(x_r)$ 的大小。 $f(\cdot)$ 其根据实际需求定，对适应度函数的选取，将在后文详细阐述。

$$fitness=\frac{1}{n}\sum_{i=1}^{n}(d_{fi}-d_{vi})^2 \tag{4}$$

其中： d_{fi} 为第 i 个样本模型输出值； d_{vi} 为第 i 个样本实际值。

更新天牛的位置，比较左右触须适应度值的大小，若 $f(x_l)>f(x_r)$ ，天牛向左移动；反之，向右移动。其更新公式如下：

$$x^{t+1}=x^t-\xi_t*\vec{d}*sign(f(x_l-x_r)) \tag{5}$$

其中 ξ_t 表示 t 次迭代步长因子； $sign(\cdot)$ 为符号函数。

1.2　性能分析

天牛须搜索算法的优化过程，主要是依靠天牛两触须对左右两端的食物气味进行辨别，哪边触须接收到的食物气味浓郁则天牛就向哪边移动，虽能大大加快天牛位置的更新和提升寻优速度，但也会导致天牛陷入局部最优，而天牛又没有跳出局部最优的能力，因此 BAS 算法在每次寻优的结果都有不同。针对高维问题，由于 BAS 算法是单个体搜索算法，单个天牛对食物气味浓度的分辨能力不足以让其在高维空间中找到最优解，这就可能导致天牛位置不更新，完全陷入局部最优，大大降低了 BAS 在高维空间的搜索精度。

2　天牛须搜索算法的改进

2.1　算法改进

本文为解决 BAS 在高维空间中优化精度低的情况，引入了二次插值算子式(6)。二次插值法是一种用于在确定初始区间中搜索极值点的方法，是一种曲线拟合方法。其利用目标函数在若干点的信息构成一个与目标函数值相近的低次多项式，再用此式的最优解作为函数的近似最优解，随着区间的逐步缩短，多项式的最优解与原函数的最优点之间的距离逐步减小，最后满足一定精度要求为止。其原理图如图 2 所示。

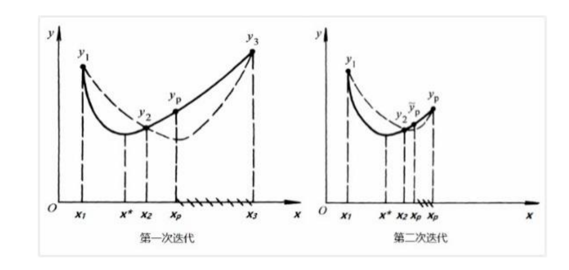


图 2　二次插值原理

Fig. 2　Principle of quadratic interpolation

本文在每一次天牛位置更新时，将左右两须所在的点作为二次插值的区间，构成一个与目标函数值相近的低次多项式，利用该式求出最优解，根据最优解再缩小区间，迭代进行，最后再满足一定精度的情况下，得到一个区间最优解，即区间最优位置。并与全局最优解进行对比，取两者最优为新全局最优解。利用这种方法大大提升了天牛须搜索算法的局部搜索能力，进而提高了天牛在高维空间中跳出局部最优的能力、搜索精度和收敛性能。

设天牛的第 t 代在高维空间中质心位置为 $X_t=(x_{t1},x_{t2},\cdots x_{tK})^T$ ，当前天牛两须位置坐标分别为 $X_l=(x_{l1},x_{l2},\cdots x_{lK})^T$ 和 $X_r=(x_{r1},x_{r2},\cdots x_{rK})^T$ ，当前全局最优位置为 $X_b=(x_{b1},x_{b2},\cdots x_{bK})^T$ ，则二次插值生成的位置为

$$x_{ik}=\frac{1}{2}\frac{(x_{lk}^2-x_{bk}^2)f(x_r)+(x_{bk}^2-x_{rk}^2)f(x_l)+(x_{rk}^2-x_{lk}^2)f(x_b)}{[(x_{lk}-x_{bk})f(x_r)+(x_{bk}-x_{rk})f(x_l)+(x_{rk}-x_{lk})f(x_b)]+eps}$$

$$k=1,2,\cdots,K \tag{6}$$

其中： $f(\cdot)$ 为适应度函数； eps 是个非常小的正数，为防止分母为 0。

QIBAS 算法的具体步骤如下：

a) 初始化天牛。根据式(1)随机生成天牛的初始朝向 $\vec{d_0}=(d_1,d_2,\cdots d_K)^T$ ，随机生成天牛的初始质心位置 $X_0=(x_1,x_2,\cdots x_K)^T$ ，并根据式(4)求得适应度值为 $f(X_0)$ ，全局最优值 $X_{best}=X_0$ ，适应度值为 $f(X_{best})$ ，设定步长因子为 ξ ，递减因子 eta ， 令 $t=0$ 。

b) 计算天牛两须坐标和适应度值。根据当前天牛的位置确定天牛两须的坐标，如式(3)所示，计算出左右触须位置坐标分别为 $X_t^l=(x_{t1}^l,x_{t2}^l,\cdots,x_{tK}^l)^T$ 和 $X_t^r=(x_{t1}^r,x_{t2}^r,\cdots,x_{tK}^r)^T$ $t=1,2,\cdots n$ ，其中 n 为总迭代次数。根据式(4)求得两须的适应度值 $f(X_t^l)$ 和 $f(X_t^r)$ 。

c) 二次插值和天牛位置更新。根据式(5)来天牛移动的位置坐标 X_t ，并求出此时质心位置对应的适应度值 $f(X_t)$ 。根据式(6)，求得插值产生的位置坐标为 $X_t^q=(x_{t1}^q,x_{t2}^q,\cdots,x_{tK}^q)^T$ ，并计算出适应度值为 $f(X_t^q)$ 。对比 $f(X_t^q)$ 和 $f(X_t)$ ， 若 $f(X_t^q)<f(X_t)$ 则更新天牛位置坐标，反之，则保持当前位置。

d) 更新全局最优值 X_{best}。对比 $f(X_i)$ 和 $f(X_{best})$，取两者中最优值替换全局最优值，更新 $f(X_{best})$ 和 X_{best}。

e) 若达到算法终止条件，则停止，否则 $t=t+1$，返回步骤 b)。

2.2 改进算法时间复杂度分析

本文的改进主要是在天牛位置更新后，利用二次插值产生一个异于当前位置的点，最后对比当前位置，取两者优值。优化问题维度为 K 时，本文算法计算复杂度分析如下：初始化天牛的计算复杂度为 $O(K)$；迭代过程，计算两须坐标和适应度值，计算复杂度为 $O(2K)$，更新天牛位置和计算适应度值，计算复杂度为 $O(2K)$，二次插值产生新位置，计算复杂度为 $O(2K)$。每一次迭代的计算复杂度为 $O(6K)$，高于标准天牛须算法的 $O(4K)$。计算复杂度可近似为 $O(K)$。表 1 列举了遗传算法(GA)、粒子群算法(PSO)、标准天牛须算法(BAS)及本文改进算法(QIBAS)的计算复杂度。表中 N 为种群数目。

表 1 算法复杂度

Tab. 1 Algorithm complexity

算法	计算复杂度
GA	$O(NK+N^2)$
PSO	$O(NK+N)$
BAS	$O(K)$
QIBAS	$O(K)$

从上表可以看出，算法计算复杂度与问题维度和种群数量有关，根据大 O 的概念，若优化问题维度 K 远大于种群数目 N 时，GA 和 PSO 两者的计算复杂度均为 $O(NK)$，而 BAS 和 QIBAS 的计算复杂度为 $O(K)$。

2.3 改进算法参数分析

根据上述算法模型可知，改进天牛须搜索算法需要设定的参数有：初始步长 ξ_0，步长递减因子 eta，总迭代次数 n，空间维度 K。对于绝大多数应用问题可根据需要设定初始步长 ξ_0，本文取值为 1。通过单因素数值实验测试函数分析改进算法其他 3 个参数对算法性能的影响。测试函数如表 2 所示。

表 2 测试函数

Tab. 2 Test function

函数名称	表达式
Sphere	$\sum_{i=1}^{D} x_i^2$
Rosenbrock	$\sum_{i=1}^{D} [100(x_{i+1}-x_i^2)^2+(1-x_i^2)^2]$
Rastrigin	$\sum_{i=1}^{D} [x_i^2-10\cos 2\pi x_i+10]$

算法参数的经验设置见表 3，在进行单因素实验时，固定其他参数，改变一个参数进行仿真实验。

表 3 经验参数设置

Tab. 3 Experience parameter setting

算法参数	变量名	取值
步长递减因子	eta	0.9
迭代次数	n	1000
空间维度	K	10

a) 步长递减因子对算法性能的影响

根据常用递减因子取值区间 $eta=[0,1]$，设置步长因子为 $eta=0.1,0.2,0.3,0.4,\cdots 0.9,1$。对测试函数进行独立 20 次数值实验。实验结果如表 4 所示。

步长递减因子的设定是为了算法在不同时期有不同的搜索半径，在算法前期搜索范围很大，需要更大的搜索半径，用来加快收敛速度。随着迭代次数的不断增加，求解值也不断逼近最优值，此时搜索半径不需要太大，需要更小的搜索范围来确定最优值。

而根据表 4 得出，对 Sphere 函数而言当递减因子 $eta=0.1$ 时算法性能最优。对 Rosenbrock 函数而言当递减因子 $eta=0.9$ 时算法性能最优。对 Rastrigin 函数而言当递减因子 $eta=0.1$ 时算法性能最优。最后本文取 $eta=0.1$。

表 4 递减因子的对算法性能影响

Tab. 4 Effect of decreasing factor on algorithm performance

函数	递减因子	最优解	最差解	平均收敛代数
Sphere	0.1	0	0	11.4
	0.2	0	0	13.8
	0.3	0	0	16.5
	0.4	0	0	20.5
	0.5	0	0	24.3
	0.6	0	0	31.1
	0.7	0	0	42.6
	0.8	0	0	59.8
	0.9	0	0	116.4
	1.0	0	0	807.5
Rosenbrock	0.1	10	10	8.2
	0.2	10	10	10.4
	0.3	9.99	10	12.5
	0.4	9.95	9.99	16.2
	0.5	9.62	9.98	20.3
	0.6	9.92	10.0	26.1
	0.7	7.68	9.88	36.5
	0.8	5.12	9.62	57.7
	0.9	4.04	8.23	130.6
	1.0	4.97	7.97	853.2
Rastrigin	0.1	-90	-90	8.5
	0.2	-90	-90	11.3
	0.3	-90	-90	14.5
	0.4	-90	-90	17.6
	0.5	-90	-90	21.7
	0.6	-90	-90	28.9
	0.7	-90	-90	38.4
	0.8	-90	-90	59.8
	0.9	-90	-90	108.6
	1.0	-86	-81	337.6

b) 最大迭代次数对算法性能的影响

最大迭代次数分别取 $n=1000,2000,3000,4000$。对测试函数进行独立 20 次数值实验。实验结果如表 5 所示。

表 5 最大迭代次数的对算法性能影响

Tab. 5 Effect of maximum iterations on algorithm performance

函数	迭代次数	最优解	最差解	平均收敛代数
Sphere	1000	0	0	116.2
	2000	0	0	115.5
	3000	0	0	117.6
	4000	0	0	116.9
Rosenbrock	1000	4.96	8.01	130.6
	2000	4.95	7.89	130.8
	3000	5.28	8.12	130.5
	4000	5.06	7.88	131.5
Rastrigin	1000	-90	-90	129.9
	2000	-90	-90	130.1
	3000	-90	-90	129.6
	4000	-90	-90	129.9

由上表可知，最大迭代次数对于本文算法的影响不大，综合考虑本文取 $n=2000$。

3 仿真测试与结果分析

3.1 参数设置

本文基于 Matlab2018b 平台仿真测试并分析 QIBAS 算

法对测试函数的计算性能。BAS 和 QIBAS 的初始步长 ξ_0 设置为 1，递减因子分别取为 $eta_1=0.9$, $eta_2=0.1$，总迭代次数为 $n=2000$，优化问题维度分别设为 $K=100, 500, 1000$，共运行 50 次测试。GA 和 PSO 的种群数量均设为 200，PSO 中学习因子均设为 1。

3.2 测试函数

为验证算法的性能，本文选取单峰和多峰等共 11 个标准测试函数[20]进行测试，测试函数如表 6、7。

3.3 实验结果与分析

将 QIBAS 算法与 GA 算法、PSO 算法以及 BAS 算法进行对比测试，各运行 50 次，取平均值，并计算标准差，实验结果如表 8(维度 $K=200$)。同时，对计算结果进行 Friedman 非参数统计分析，其中显著性水平设为 5%，实验结果如表 9 所示。表 10 是各算法运行 50 次 f_1、f_6、f_{11} 三个函数所需的时间。

从表 8 可以看出(最优值已加粗)，在 50 次的函数测试中，QIBAS 算法在对 f_1、f_2、f_3、f_4、f_{11} 函数的计算都收敛到了最小值 0，标准差也为 0，证明了算法的稳定性较高。其他的如 f_7、f_8、f_9、f_{10} 4 个函数虽没有收敛到最小值，但与其他算法相比其收敛效果都要好。而 f_5、f_6 2 个函数与 PSO 相比收敛效果还是较差的，但比 GA 算法效果更优。在与标准 BAS 算法比较时，除了 f_5、f_6、f_{10} 三个函数的计算结果相差不大，其余函数计算结果的平均值和标准差都较优。由此可以看出改进天牛须搜索算法的计算精度和稳定性都有较明显的提升，验证了改进算法寻优的有效性。

表 6 单峰函数

Tab. 6 Unimodal function

函数	测试函数	函数	测试函数
f_1	Sphere	f_5	Rosenbrock
f_2	Schwefel Problem 2.22	f_6	Step
f_3	Schwefel Problem 1.2	f_7	Quartic
f_4	Schwefel Problem 2.21	f_8	Schwefel Problem 2.26

表 7 多峰函数

Tab. 7 Multimodal function

函数	测试函数
f_9	Rastrigin
f_{10}	Ackley
f_{11}	Griewank

表 8 算法测试结果对比

Tab. 8 Comparison of algorithm test results

函数	GA		PSO		BAS		QIBAS	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
f_1	25.9571	1.3570	0.0823	0.0322	52.9531	3.9961	**0**	**0**
f_2	47.7552	3.2680	1.4682	0.5800	84.8440	3.6806	**0**	**0**
f_3	8.62E+04	9.35E+03	4.47E+02	8.12E+03	9.69E+05	5.88E+06	**0**	**0**
f_4	0.7854	0.0233	0.0455	0.8096	0.9957	3.40E-03	**0**	**0**
f_5	3.499E+02	19.7682	**8.65E-07**	**1.40E-06**	7.00E+02	11.9674	2.01E+02	7.08E-07
f_6	2.00E+02	0	**0**	**0**	99.80	6.4175	99.22	5.9395
f_7	5.65E+04	3.35E+02	2.61E+03	9.5392	2.82E+03	4.30E+02	**3.29E-04**	**3.15E-04**
f_8	-81.1058	1.2893	-1.69E-02	9.13E-04	-22.7833	7.2362	**-1.33E+03**	**7.84E+03**
f_9	9.91E+02	30.1911	1.5973	0.9359	-6.25E+02	58.9376	**-1.99E+03**	**0**
f_{10}	-4.0824	0.0231	-4.7104	3.90E-03	-20.2773	0.1413	**-21.7376**	**1.3725**
f_{11}	0.1748	7.91E-03	5.62E-04	2.63E-04	1.0136	1.12E-03	**0**	**0**

从表 9 的结果可以看出，四个算法中本文改进算法的秩均值最小，由此得出 QIBAS 的性能强于其他三种算法。

从表 10 可以看出，不论单峰函数或者多峰函数，QIBAS 算法和 BAS 算法的运行时间远小于 PSO 算法和 GA 算法，时间最高相差达到 100 多倍。QIBAS 与 BAS 两个算法的运行时间相差不多，QIBAS 所需时间要长一些。PSO 和 GA 各有优劣。因此 QIBASh 和 BAS 运行所需时间要小的多，算法性能更优。

为了进一步检验二次插值对 BAS 算法带来的效果，分别进行天牛须结合模拟退火(SABAS)、变步长天牛须(VSBAS)、天牛群算法(BSAS)，共进行 50 次测试，测试结果如表 11。

由表 11 的计算平均值和标准差可知，SABAS 算法和 VSBAS 相比于标准 BAS 算法其性能都有不同程度的提升，但其效果并不太明显。QIBAS 算法相比于 SABAS 算法除了 f_5、f_6，其他函数的优化效果明显 QIBAS 远高于 SABAS。而相比于 VSBAS 算法，QIBAS 在 f_6、f_{10} 函数的计算结果稍差，其余函数均优于 VSBAS 算法。大部分测试结果优于 BSAS 算法。

从表 12 的检验来看，本文算法的秩均值比 SABAS、VSBAS 和 BSAS 等改进算法都要小，因此性能要更优。

结合表 8 和 10 可以看出，50 次的测试结果，所得的标准差中，本文改进算法相较于其他算法，明显降低了，由此证明了本文算法的稳定性和可靠性。

为了更直观的展现各算法在测试函数上的收敛情况，选取结果与 50 次实验平均值最为接近的实验，同时对比 VSBAS 算法、SABAS 算法、BSAS 算法以及 BAS 算法，绘制收敛曲线。选取 f_1、f_4、f_9、f_{11} 四个测试函数在空间维度 $K=200$ 进行收敛曲线对比，如图 3~6。图 7 是 $K=500$ 时 f_1、f_9 函数的收敛曲线。图 8 是 $K=1000$ 时 f_1、f_9 函数的收敛曲线。

表 9 Friedman 检验

Tab. 9 Friedman test

算法	秩均值	算法	秩均值
QIBAS	1.64	BAS	3.25
PSO	2.55	GA	3.49

表 10 算法运行时间

Tab. 10 Algorithm running time

算法	f_1 时间/s	f_6 时间/s	f_{11} 时间/s
QIBAS	2.860	5.476	7.766
PSO	135.736	169.136	278.286
BAS	1.540	2.302	3.416
GA	189.886	241.669	160.109

表 11　算法测试结果对比

Tab. 11　Comparison of algorithm test results

函数	BAS		SABAS		VSBAS		BSAS		QIBAS	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
f_1	59.1806	3.9961	41.6044	3.9460	55.5218	3.5925	4.93E-30	0	**0**	**0**
f_2	84.8440	3.6806	77.2043	3.5957	99.3674	3.9986	2.22E-15	2.6354	**0**	**0**
f_3	9.69E+05	5.88E+06	9.46e+04	7.25e+04	2.47e+02	3.91e+03	2.32E-41	2.68E+02	**0**	**0**
f_4	19.7657	3.40E-03	19.1583	0.2346	17.6290	1.12E-02	1.57E-15	1.0238	**0**	**0**
f_5	7.00E+02	11.9674	45.5127	9.73E+02	1.04e+04	1.09e+03	**0.9899**	**9.1165**	2.01E+02	7.08E-07
f_6	99.80	6.4175	**94.9400**	**6.8436**	98.4400	6.0243	730	2.96E+02	99.22	5.9395
f_7	2.82E+03	4.30E+02	1.73e+03	2.97E+02	4.09E+02	4.30E+02	**3.37E-05**	**2.15E-02**	3.29E-04	3.15E-04
f_8	-22.7833	7.2362	-35.7983	6.7151	-21.4780	6.9626	-87.7179	6.8965	**-1.33E+03**	**7.84E+03**
f_9	-6.25E+02	58.9376	-9.05E+02	56.9366	-88.1146	61.7779	3.9798	8.21E+02	**-1.99E+03**	**0**
f_{10}	-20.2773	0.1413	-20.8196	0.1154	**-29.5700**	**0.4623**	-21.1639	2.3569	-21.7376	1.3725
f_{11}	7.5749	1.12E-03	6.2596	8.25e-04	8.0248	9.02e-04	10.2728	3.21E+02	**0**	**0**

表 12　Friedman 检验

Tab. 12　Friedman test

算法	秩均值	算法	秩均值
QIBAS	1.64	VSBAS	2.86
BSAS	2.26	BAS	3.25
SABAS	2.77		

从图 3~6 可知，在维度 $K=200$ 时，对于不同函数 BAS、BSAS 算法和 SABAS 算法的收敛效果各有优劣，而 VSBAS 算法的收敛效果一直优于 BAS 算法，但与本文改进的算法相比，效果并不显著。从图中可以看出 QIBAS 算法的收敛效果远远高于其他四种算法。

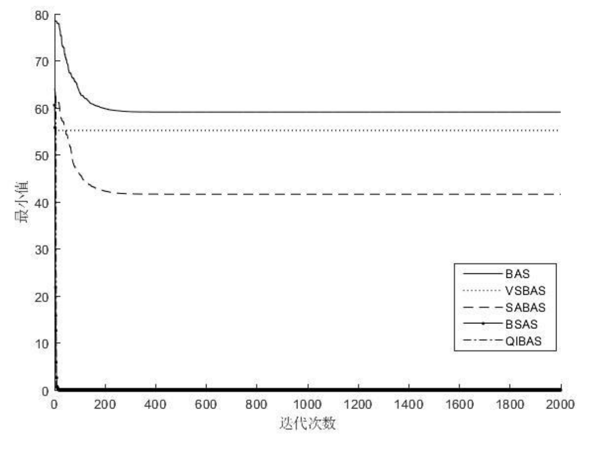


图 3　函数 f_1 收敛性能

Fig. 3　Function f_1 convergence performance

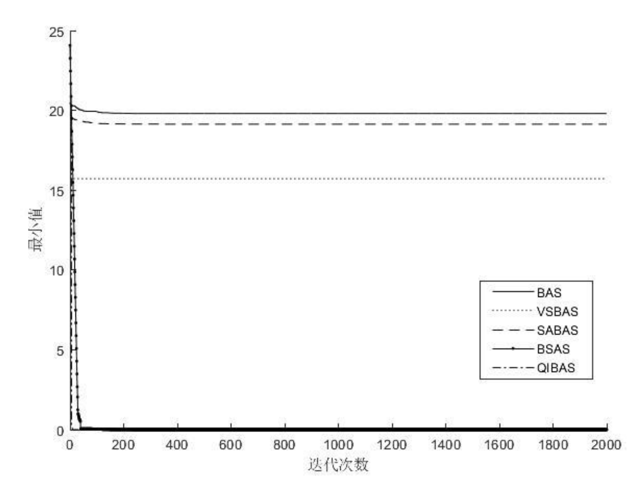


图 4　函数 f_4 收敛性能

Fig. 4　Function f_4 convergence performance

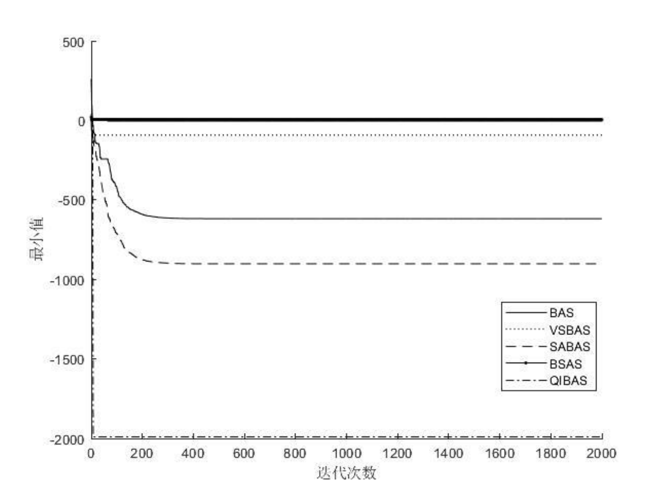


图 5　函数 f_9 收敛性能

Fig. 5　Function f_9 convergence performance

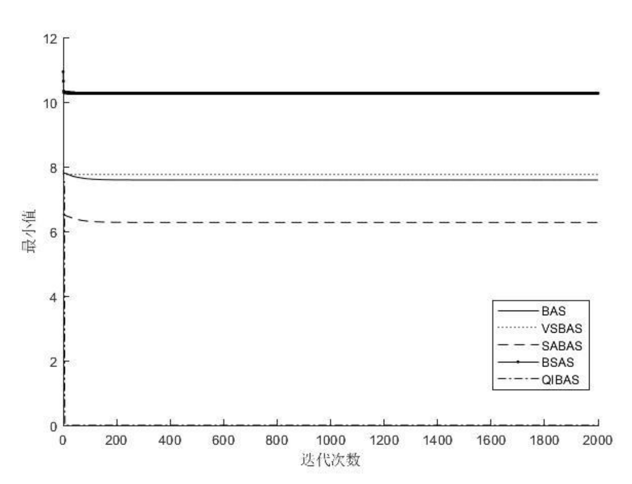
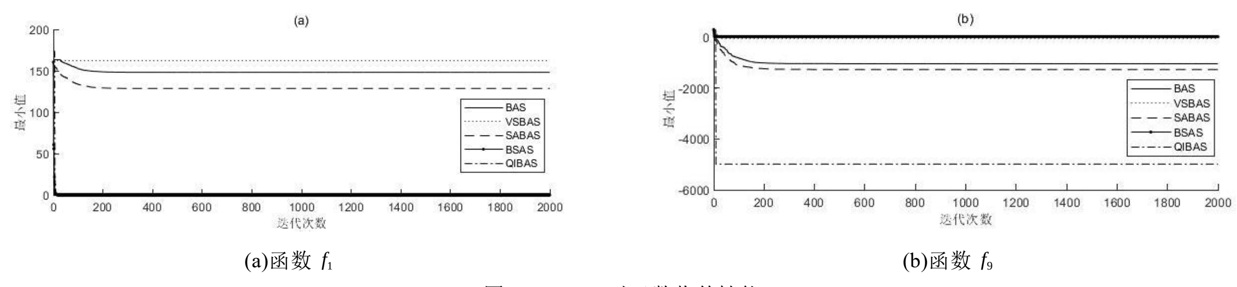


图 6　函数 f_{11} 各算法收敛性能对比

Fig. 6　Function f_{11} convergence performance

从图 7 和 8 可知，BAS 算法、VSBAS 算法和 SABAS 算法在维度分别为 $K=500$ 和 $K=1000$ 时，基本没有太大的收敛效果，甚至不收敛，而 BSAS 算法只是对函数 f_1 的收敛效果较好。这也证实了 BAS 算法在高维空间搜索的局限性。但本文改进的算法却依旧有较好的收敛性，甚至在 $K=1000$ 时，针对 f_9 函数的收敛效果比在 $K=500$ 的最优值更接近实际最优值。

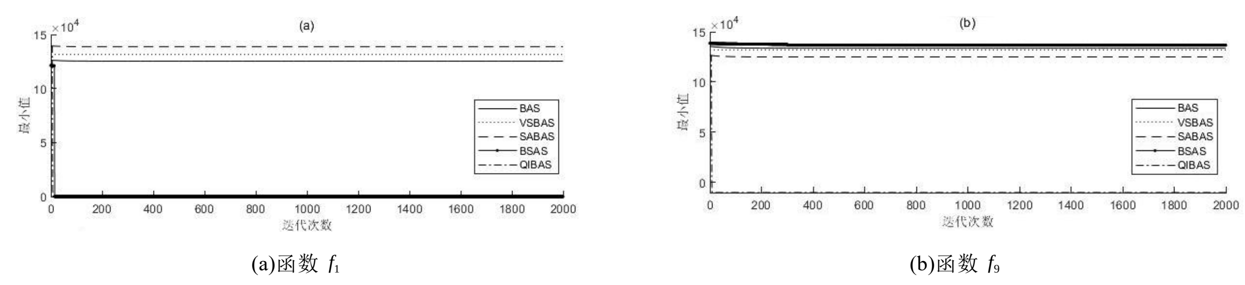
为了更进一步验证在更高维的问题中，本文算法的可行性，设计空间维度 $K=5000,10000$，其结果如图 9 和 10 所示。从图中可以看出在维度为 5000，10000 时，本文算法依旧有更好的寻优性能。



(a)函数 f_1 (b)函数 f_9

图 7 K=500 时函数收敛性能

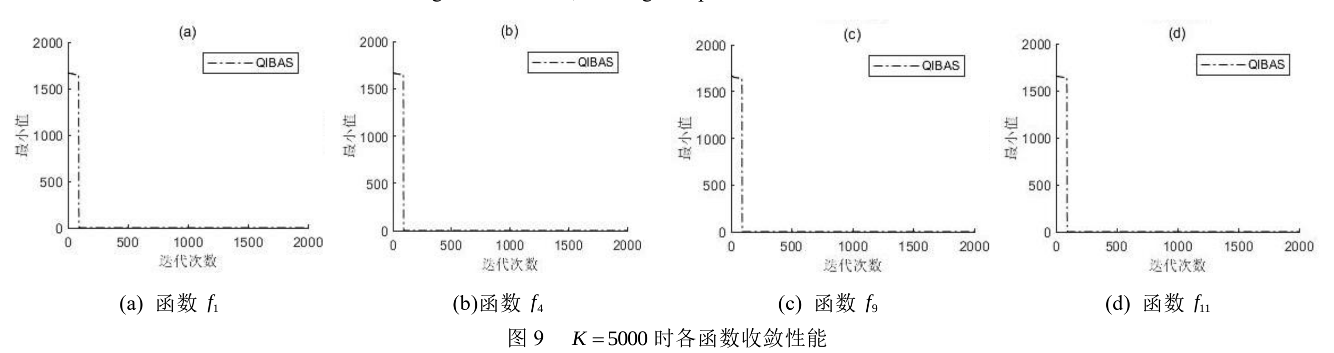
Fig. 7 $K=500$, convergence performance of functions



(a)函数 f_1 (b)函数 f_9

图 8 K=1000 时函数收敛性能

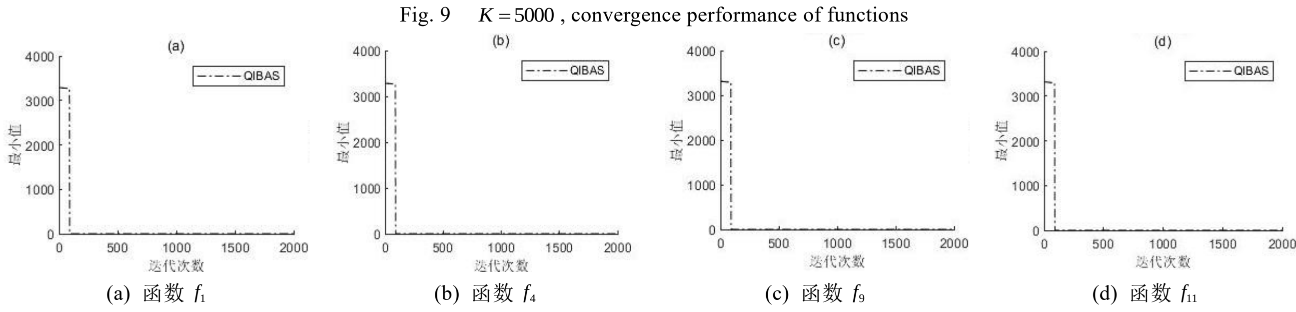
Fig. 8 $K=1000$, convergence performance of functions



(a) 函数 f_1 (b)函数 f_4 (c) 函数 f_9 (d) 函数 f_{11}

图 9 $K=5000$ 时各函数收敛性能

Fig. 9 $K=5000$, convergence performance of functions



(a) 函数 f_1 (b) 函数 f_4 (c) 函数 f_9 (d) 函数 f_{11}

图 10 $K=10000$ 时各函数收敛性能

Fig. 10 $K=10000$, convergence performance of functions

综合上图可以得出结论，本文改进算法(QIBAS)在维度不断加高时，依旧有较高的收敛精度，还有极快的收敛速度。而且在针对不同函数时，本文改进算法的收敛效果均较好，也侧面突出了改进策略的稳定性。而其他四种算法却随着维度的加高，针对不同函数来说，有些函数存在收敛效果，但对比低维度时效果大大减弱，收敛速度也大大降低。而有些则存在不收敛的情况。这也更进一步验证改进策略的稳定性和有效性。

4 结束语

为解决天牛须搜索算法在高维空间中搜索精度低和易陷入局部最优的问题，本文提出一种基于二次插值的天牛须搜索算法。通过二次插值的方式提升天牛跳出局部最优的能力，也有效的提升了天牛在高维空间中的搜索精度。仿真实验里对 11 个测试函数进行计算，结果表明，相比于遗传算法、粒子群算法和标准天牛须算法，改进算法在计算精度、稳定性和收敛速度上有明显的提升。同时也比较了引入模拟退火的天牛须算法、变步长策略的天牛须算法以及天牛群搜索算法，统计分析和计算结果均证明本文改进策略的有效性和稳定性。

参考文献：

[1] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents [J]. IEEE Trans on Systems Man & Cybernetics Part B Cybernetics A, 1996, 26 (1): 29.

[2] Kennedy J, Eberhart R. Particle swarm optimization [C]// Proc of IEEE International Conference on Neural Networks. Perth, Australia, USA: IEEE Press, 1995: 1942-1948.

[3] Coit D. Genetic algorithms and engineering design [J]. Engineering Economist, 1998, 43 (4): 379-381.

[4] Karaboga D. An idea based on honey bee swarm for numerical optimization: Technical Report-TR06 [R]. Kayseri: Erciyes University, 2005.

[5] Mirjalili S, Lewis A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[6] Deng Wu, Xu Junjie, Zhao Huimin. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem [J]. IEEE access, 2019, 7: 20281-20292.

[7] Nouiri M, Bekrar A, Jemai A, *et al.* An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem [J]. Journal of Intelligent Manufacturing, 2018, 29 (3): 603-615.

[8] Abdelhalim H, Ali D, Iyad R. A genetic algorithm approach for location-inventory-routing problem with perishable products [J]. Journal of Manufacturing Systems, 2017, 42.

[9] Wang Wenjie, Tian Guangdong, Chen Maoning, *et al.* Dual-objective program and improved artificial bee colony for the optimization of energy-conscious milling parameters subject to multiple constraints [J]. Journal of Cleaner Production, 2020, 245.

[10] 褚鼎立，陈红，王旭光. 基于自适应权重和模拟退火的鲸鱼优化算法 [J]. 电子学报, 2019, 47 (05): 992-999. (Chu Dingli, Chen Hong, Wang Xuguang. Whale optimization algorithm based on adaptive weights and simulated annealing [J]. Acta Electronica Sinica, 2019, 47 (05): 992-999.)

[11] 肖理庆，王化祥. 利用改进粒子群算法优化 ERT 有限元模型 [J]. 计算机应用研究, 2017, 34 (05): 1581-1584. (Xiao Liqing, Wang Huaxiang. Optimization of ERT finite element model using improved particle swarm optimization [J]. Application Research of Computers, 2017, 34 (05): 1581-1584.)

[12] Jiang Xiangyuan, Li Shuai. BAS: beetle antennae search algorithm for optimization problems [J]. arXiv preprint arXiv: 1710. 10724, 2017.

[13] Zhu Zongyao, Zhang Zhiyu, Man Weishi, *et al.* A new beetle antennae search algorithm for multi-objective energy management in microgrid [C]// Pro of the 13th IEEE Conference on Industrial Electronics and Applications (ICIEA) . IEEE, 2018: 1599-1603.

[14] Sun Yuantian, Zhang Junfei, Li Guichen, *et al.* Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes [J]. International Journal for Numerical and Analytical Methods in Geomechanics, 2019, 43 (4): 801-813.

[15] Jiang Xiangyuan, Li Shuai. Beetle antennae search without parameter tuning (BAS-WPT) for multiobjective optimization [J]. arXiv preprint arXiv: 1711. 02395, 2017.

[16] Wang Jiangyu, Chen Huanxin. BSAS: Beetle Swarm Antennae Search Algorithm for Optimization Problems [J]. arXiv preprint arXiv: 1807. 10470, 2018.

[17] 卢光辉，滕欢，廖寒逊，等. 基于改进天牛须搜索算法的分布式电源选址定容 [J]. 电测与仪表, 2019, 56 (17): 6-12. (Lu Guanghui, Teng Huan, Liao Hanxun, *et al.* Location and volume of distributed power supply based on improved beetle search algorithm [J]. Electrical Measurement and Instrumentation, 2019, 56 (17): 6-12.)

[18] Chen Tingting, Zhu Yongjian, Teng Jun. Beetle swarm optimization for solving investment portfolio problems [J]. The Journal of Engineering, 2018, 2018 (16): 1600-1605.

[19] 孔慧华，孙英博，张雁霞. 基于天牛须搜索的全变分最小化算法在计算机断层成像内重建中的应用 [J]. 激光与光电子学进展, 2019, 56 (21): 90-96. (Kong Huihua, Sun Yingbo, Zhang Yanxia. The application of the full variational minimization algorithm based on the beetle whisker search in the reconstruction of computer tomography [J]. Las Optoelect Prog, 2019, 56 (21): 90-96.)

[20] Fang Wei, Sun Jun, Chen Huanhuan, *et al.* A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population [J]. Information Sciences, 2016, 330 (C): 19-48.